ED-cPSD Documentation

Andre Adam, Guang Yang, Huazhen Fang, Xianglin Li

Last Updated: March 6, 2025

Contents

Introduction

This document serves as reference and documentation for ED-cPSD, a software based on a novel algorithm of sequential erosion-dilation for continuous phase-size distribution (ED-cPSD) estimation. This document is segmented into a few different chapters which serve as reference for both users and contributors.

Chapter 1 is dedicated to get new users running the software and generating meaningful results immediately. Chapter 1 will walk new users through the installation process, basic GUI usage, running code de-coupled from the GUI, preparing inputs with ImageJ [1] and/or simple python scripts (OpenCV), expected outputs, and interpreting outputs.

Chapter 2 will first present an in-depth mathematical description to the algorithm working behind the scenes to calculate the continuous phase-size distribution (cPSD). This chapter also includes implementation details for both the standalone code and the GUI computational model. These implementation details are important for someone who is looking to contribute to and/or fork this project.

Chapter 3 presents several examples and case studies of the use of this software/algorithm. Each case study carries highlights quintessential aspects of the algorithm, such as discussions on how to interpret the output data given different inputs, comparisons with other cPSD and discrete phase-size distribution (dPSD) algorithms, and use cases in materials science and energy research.

Finally, chapter 4 has additional information, such as acknowledgments, code licensing information, and references.

Chapter 1

Quick-Start Guide

This chapter will be a brief introduction on how to download the software GUI and produce results with minimal setup required. This chapter will also mostly focus on Windows users. Some special instructions for Linux users are included in a separate section.

1.1 Setup

1.1.1 Method 1: Download Executable

The first step for getting the software running is to download it directly from one of the source repositories. For users that are inexperienced with Git, it is recommended to go directly to the project GitHub repository at `https://github.com/adama-wzr/ED_cPSD`. From there, select the green "Code" button, and select the option "Download Zip", as shown in figure 1.1.

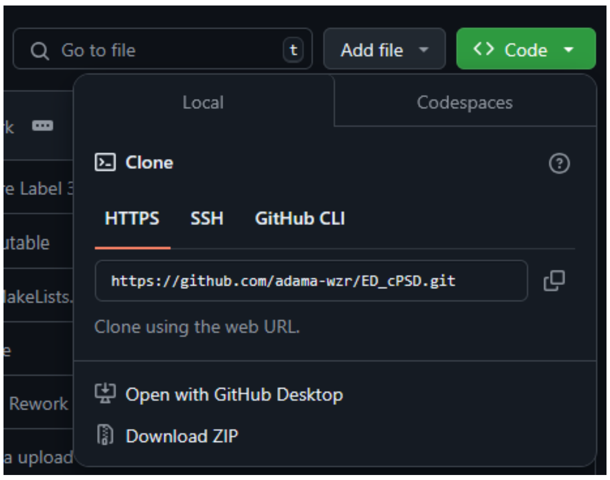


Figure 1.1: Select the green button, and then select the option to download the zip containing all files.

Once the zip is downloaded, it can be extracted to the desired location in the computer. The executable can be found in the folder "ED_cPSD/build/Desktop_Qt_6_8_1_MinGW_64_bit/". The executable is titled "ED_cPSD.exe". You can right click the ".exe" and create a shortcut, and then copy the shortcut to a place where it is more accessible, or just search for the executable using Windows search.

Verify at this stage that you are able to open the software. If you can open it, the everything should be running. Skip to one of the examples to learn how to use the software.

1.1.2 Method 2: Build Project

Statically linking libraries is relatively memory consuming on a computer, and might not be the best way to build the software for the experienced user. Thus, some may instead choose to build the executable.

In order to do that, just make sure the following requirements are met:

- Download the "GUI_src" folder from GitHub.
- Ensure a modern C++ version is installed.
- Ensure you have Qt6.8 installed and it is in the PATH (any recent major release of Qt should work, but only 6.8 was tested).
- (optional) Qt Creator 15.0 or newer (other versions might work).

Now, on the Qt creator, select the option to "Open Project", navigate to the folder where the "GUI_src" was saved, and select the file "CMakeLists.txt". A window will then appear asking the user to configure the project, but no additional configurations are need. Simply select the "Configure Project" button on the right side of this window, and Qt will build it for you.

At this stage, the GUI should be ready to run within the Qt Creator, simply press "Run" or "Ctrl-R". The software outside of the Qt Creator if Qt is already in the PATH environment variables. A simpler alternative is to use the Qt Deployment Tool for Windows `https://doc.qt.io/qt-6/windows-deployment.html`, which will statically link the needed libraries from Qt directly to the executable file's folder.

1.2 Examples

Also in the code repository, there is a folder labeled "Examples", with three simulation examples (one in 2D and two in 3D). In this folder, the input files and expected outputs are already included, but in this section we will examine how to reproduce those examples.

Make sure the source files containing the structure information (`.csv` and `.jpg`) are saved at your computer. The input files and simulations will be run using the GUI, and that can be installed anywhere on the computer.

1.2.1 2D Example

This example will show you how to process an image using ImageJ and properly format it to be an acceptable input to the code. The image in question is `cRec2000.jpg`, a 2000×2000 reconstruction of a carbon electrode, featured in Adam 2022 [2].

The first step is to open the image using ImageJ. If the image is not of `.jpg`, the **FIRST** step is to go to the "File" tab, select "Save As", and then save as `.jpg`.

Once the image is saved as `.jpg`, go to the "Image" tab, select the option "Type", and make sure the image is saved as a "8-bit". If there is a need for thresholding, go to the "Image" tab, "Adjust", and then "Threshold". Here the threshold can be adjusted. The software always assumes that white is pore, and black is particle. The image can either be converted to binary here, or the thresholding value to separate particle from pore can be entered into ED-cPSD directly. Remember to save again after these modifications are done.

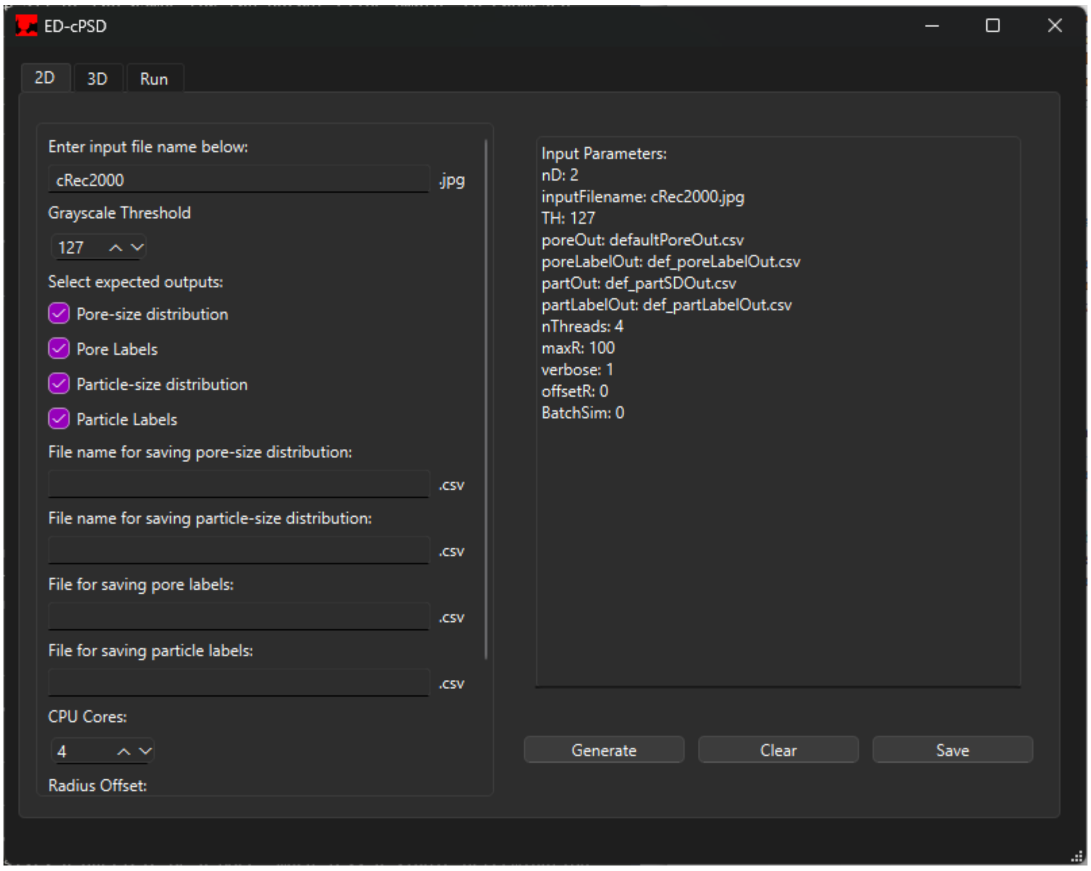
Once the image is saved, you can check that the process worked by right clicking on it, select "Properties", then "Details", and finally just verify that under bit-depth it says 8-bit. That indicates a single channel grayscale image. An RGB image will usually say 24 or 32-bit, and we don't want that.

Now, we have to generate the input file for the simulation. Open the ED_cPSD software, and navigate to the 2D tab if necessary. Under the 2D tab, the very first field asks for the image name, which in this example is "cRec2000". Do not enter the file extension, `.jpg`, as that is already entered automatically.

Beneath the file name, the threshold can be adjusted on a range from 1 to 255. Below that, the user can select the desired outputs. For the sake of the example, we will select all of the options. Also, we will leave all of the names for the output files empty, to showcase that defaults will be set in place if a user doesn't select a name.

The last options relate to simulation parameters. The number of threads simply controls the number of CPU threads OpenMP will pool for the simulation. The radius offset can be used for smoothing, it means the simulation will start at a radius of $r = r_{\text{off}}$ instead of $r = 1$. The "Max. Radius" option sets a maximum radius cutoff for the simulations. Even if the solution is not yet complete, ED-cPSD will not scan beyond $r = r_{\text{max}}$. Finally, the option "Verbose" controls whether output will be printed to the window (GUI) or command line (for running the code on the command line).

Press the "Generate" button, and the input file will be displayed on the screen as seen in figure 1.2.


ED-cPSD
2D
3D
Run
Enter input file name below:
cRec2000
.jpg
Grayscale Threshold
127
Select expected outputs:
Pore-size distribution
Pore Labels
Particle-size distribution
Particle Labels
File name for saving pore-size distribution:
.csv
File name for saving particle-size distribution:
.csv
File for saving pore labels:
.csv
File for saving particle labels:
.csv
CPU Cores:
4
Radius Offset:
Generate
Clear
Save

```
Input Parameters:
nD: 2
inputFilename: cRec2000.jpg
TH: 127
poreOut: defaultPoreOut.csv
poreLabelOut: def_poreLabelOut.csv
partOut: def_partSDOut.csv
partLabelOut: def_partLabelOut.csv
nThreads: 4
maxR: 100
verbose: 1
offsetR: 0
BatchSim: 0
```

Figure 1.2: Example input file on ED-cPSD for the 2D example.

Press the “save” button, and be sure to save the input in the same folder as the image you are trying to simulate. Now go over to the “Run” tab, select the same folder again, or manually enter the path to the folder. At this stage, the simulation is ready to run, so just press the “Run” button and watch the results being printed to the text tab on the right-hand side of the screen.

Even though the image is 2000×2000, both pore and particle size distributions will be obtained within a few seconds. The GUI has a 0.25 ms delay every time it prints something to avoid pointers being lost to garbage collection, thus it may take longer to print the output than it does to actually run the simulation. If this is an issue, consider turning “verbose” off or running the code in the command line directly (no delay).

For processing the output files, see some suggestions in section 1.3, where the topic is covered. It is also important to understand the assumptions of the model and what is being calculated, and this information can be found in chapter 2.

1.2.2 3D Example: csv Input

1.3 Processing Output

The phase-size distributions are interpreted the same way whether they are in 2D or 3D, and this is addressed in subsection 1.3.1. The subsection contains information of how to transform the output to a size distribution based on diameter, include pixel resolutions, and how to get the average diameter, D_{50}.

However, plotting the particle labels by radius can be tricky in 3D, and therefore that discussion is divided into a 2D discussion in subsection 1.3.2 and 3D in subsection 1.3.3.

1.3.1 Phase-Size Distribution Output

Regardless if the data was originally in 2D or in 3D, the format of this output is always the same. The first column will have the pore radius, with the label “r”, and the second column will have the probability density of a given particle of size “r”, and this column is labeled “p(r)”.

The first column can easily be converted to the real diameter, by using equation 1.1.

$$d = 2 \cdot r \cdot pr \tag{1.1}$$

where pr is the pixel resolution of the base image.

An average diameter can be obtained from that in two ways. The simplest way to obtain D_{50} is by equation 1.2

$$\mathrm{D}_{50} = \sum_{d_{min}}^{d_{max}} d \cdot p\,(d) \tag{1.2}$$

Another way to obtain it is to calculate a cumulative probability distribution, as is done in other articles [3, 4]. This method interpolates and finds at what particle diameter the cumulative probability distribution reaches 0.5. These two methods arrive at the same D_{50}.

1.3.2 Phase-Labels in 2D

The ED-cPSD label files have 4 columns in 2D, and the columns are the x and y pixel coordinates, the radius label “R”, and the label “L”, respectively. For more information on how “R” and “L” are derived, see section 2.3.

This brief tutorial will show how to plot a colormap of “R” on the original image domain. The output can be very informative for visualizing the approximate locations and concentrations of certain particle sizes and shapes. This also provides good understanding of how the the ED-cPSD code is measuring objects.

In order to plot, any plotting algorithm can be used, but for the sake of the tutorial, we will use Python with the libraries `matplotlib`, `pandas`, and `numpy`. The file used in this tutorial is included in the GitHub folder, under `/Examples/Contour Plot 2D`, with all the necessary files also included. The code below is well commented, and very straight-forward.

The code below produces figure 1.3 using the data from subsection 1.2.1.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

plt.rcParams["font.family"] = "Times New Roman"

# image size properties
xSize = 2000
ySize = 2000
pixel_Resolution = 1  # units

# Read image
imgName = "cRec2000.jpg"
img = np.uint8(mpimg.imread(imgName))

# Read .csv
df1 = pd.read_csv("def_partLabelOut.csv")

# get raw data from csv files
raw_data = df1[["x", "y", "R", "L"]].to_numpy()
R = np.zeros((ySize,xSize))

x = raw_data[:,0]
y = raw_data[:,1]

for i in range(len(raw_data[:,0])):
 R[y[i]][x[i]] = raw_data[i,2]*pixel_Resolution
```

```
# Create the mesh grid
Xp, Yp = np.meshgrid(np.linspace(0, 1, xSize), np.linspace(1.0*ySize/xSize, 0, ySize))

# plotting
fig1, ((ax1, ax2)) = plt.subplots(1, 2, constrained_layout=True)
fig1.set_dpi(100)
fig1.set_size_inches(8, 4)

# First axis is just the image
ax1.imshow(img)
ax1.set_title(imgName, fontsize=16)

# Second axis is R - contour
CS2 = ax2.contourf(Xp, Yp, R, 40, cmap=plt.cm.rainbow)
cbar2 = fig1.colorbar(CS2, ax=ax2, fraction=0.046, pad=0.04)
cbar2.set_label(r'Particle Radius [voxels]', rotation=90, fontsize=14)
ax2.set_title("Particle Radius Distribution", fontsize=16)
ax2.set_aspect('equal', adjustable='box')
plt.show()
```

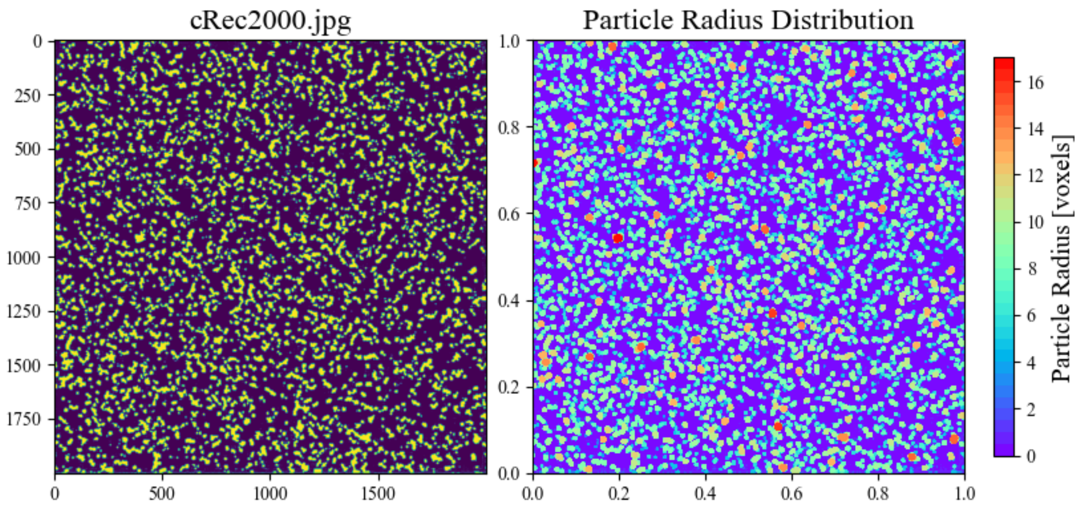


Figure 1.3: Example contour plot from processed ED-cPSD output.

1.3.3 Phase Labels in 3D

In this brief tutorial I will show another way of displaying the phase labels like in subsection 1.3.2. This can still be done using Python, but instead this tutorial will display another technique using the open-source software Paraview [5].

The ED-cPSD label files have 5 columns in 3D, and the columns are the x, y, and z pixel coordinates, the radius label "R", and the label "L", respectively. The output `.csv` file can be read directly into Paraview. Simply open the software, select "File", then "Open", and import the particle labels output from example 1.2.2. Make sure to select the "Global CSV Reader".

A menu will appear on the left-hand side of the screen. Simply click "Apply" to the default information. Once the software is done loading the information, the "SpreadSheetViewer" will appear on the screen, but it can be closed right away. Select the loaded file on the pipeline browser on the left-hand side of the screen, then select the "Filters" tab at the very top of the page, search for "TableToPoints" filter and select it.

New options open up on the left-hand side menu. Under the column, select "x" for the X column, "y" for the Y column, and "z" for the Z column. Hit "Apply". This will generate a rendering of the 3D structure, but it is still hard to identify any features or read the contours, thus several modifications are needed at this step.

On the left-hand side menu, click on the cog next to the search bar, and it will “toggle advanced options”. Make sure those options are available. Under “Representation”, change from “Surface” to “3D Glyphs”. Right below it, on the “Coloring” section, change the options from “solid” to “R”. Paraview will automatically use the radius labels “R” to apply a colormap to the surface.

Scroll down to “Glyph Parameters”, under “Scale Array”, make sure “None” is selected. Change “Glyph Type” from “Arrow” to “Box”. At this stage, a good visualization is already obtained.

Some other optional steps for improving the quality of the output include changing the background to a solid color with better contrast, adjusting the colorbar settings, and adjusting the axis and axis labels.

By following these steps (including the optional ones), we arrive at figure 1.4.

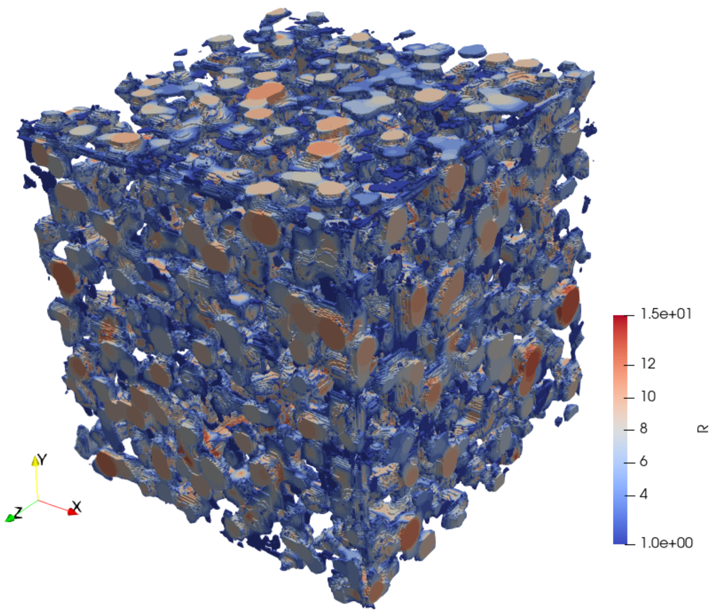


Figure 1.4: Example contour plot from processed ED-cPSD output for a 3D structure.

Chapter 2

Algorithm Details

2.1 Motivation

The interplay between different phases in heterogeneous materials [6] is of paramount importance in determining the bulk-scale properties of said materials. More specifically, in porous media, the interplay between the particle and pore spaces are critical in determining the transport properties, as has been thoroughly discussed in scattered studies[7, 8, 9, 10, 11, 12, 13].

Several software packages and algorithms already aim take aim at reconstructing, analyzing, and characterizing the morphology of such complex systems, such as MCRpy [14] and MATBOX [15]. On the other hand, there are different interpretations on the meaning and accuracy of physical descriptors.

One such property is the particle/pore/phase size distribution, which is often reported as just the average pore/particle diameter, D_{50}. This is obviously an issue, as there is often no definition of what consists a particle or a pore, much less a single determination of what "size" means without context (i.e. what is the size of an ellipsoid? The semi-major axis, the semi-minor axis, the average diameter, the length, or an average of all of those?).

In some materials, like carbon-based materials for battery electrodes or some types of sandstone can be observed under scanning electron microscopy (SEM) to be largely composed of small approximately spherical individual particles. In such cases, segmenting particles within the domain and assigning them sizes is feasible, in what is considered a discrete particle-size distribution (dPSD) algorithm.

While different dPSD algorithms have different approaches, the basic scheme is the same: each domain is segmented into particles, and each particle is labeled separately. Then, the morphology of each particle is quantified, and from there the averages can be obtained (i.e. PSD and D_{50}. More details and a brief compilation of such algorithms can be found in Usseglio-Viretta (2020) [4].

Another avenue for obtaining the PSD is via continuous pore-size distribution (cPSD) algorithms. This class of algorithms implies a continuous assessment of the size of each phase space. In other words, no segmentation is needed and there is a continuous measurement of phase-space morphology. This interpretation is more consistent when considering materials like stochastic aluminum foams [16], where both the aluminum and pore-space are continuous and amorphous, hence incompatible with dPSD.

Even in better structured materials like triply-periodic minimal surfaces (TPMS), the cPSD measurements of particles and pore size distributions are physically interpretable, representing the probability density of thickness distribution and pore-space minimum radius distribution, respectively. Or, in the aforementioned examples of carbon-based battery electrodes and sandstone, the pore-space is simply the interstitial space defined by the absence of solid particles, hence it is also continuous and amorphous. Approaching such cases with a dPSD algorithm can be tricky, as it can quickly lead to over-segmentation [4].

Furthermore, the cPSD class of algorithms more closely resemble mercury intrusion porosimetry [3] or nitrogen desorption porosimetry [17], which are experimental methods for determining pore-size distributions. Again, a brief review and further discussion can be found in Usseglio-Viretta (2020) [4].

On the other hand, the cPSD algorithms do have several downsides [4], including the necessity for data fitting, strong assumptions about particle shape (the spherical-particle assumption), the lack of information about the location and anisotropy of the related spaces, no mechanism for particle identification, requiring large representative volumes for accurate assessment, and often times underestimation of average phase sizes.

2.2 cPSD via Sequential Erosion-Dilation: ED-cPSD

The ED

2.3 Phase Labeling

Chapter 3

Case Studies

Chapter 4

Additional Information

4.1 Acknowledgments

This work used Expanse(GPU)/Bridges2(CPU) at SDSC/PSC through allocations MAT210014 and MAT230071 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by National Science Foundation grants 2138259, 2138286, 2138307, 2137603, and 2138296 [18].

X.L. highly appreciates the support from the National Science Foundation (Award 1941083 and 2329821).

The information, data, or work presented herein was funded in part by the Advanced Research Projects Agency-Energy (ARPA-E), U.S. Department of Energy, under Award Number 18/CJ000/08/08.

Bibliography

[1] Schneider, C. A., Rasband, W. S. & Eliceiri, K. W. Nih image to imagej: 25 years of image analysis. *Nature Methods* **9**, 671–675 (2012). URL http://dx.doi.org/10.1038/nmeth.2089.

[2] Adam, A., Wang, F. & Li, X. Efficient reconstruction and validation of heterogeneous microstructures for energy applications. *International Journal of Energy Research* (2022). URL https://onlinelibrary.wiley.com/doi/full/10.1002/er.8578.

[3] Münch, B. & Holzer, L. Contradicting geometrical concepts in pore size analysis attained with electron microscopy and mercury intrusion. *Journal of the American Ceramic Society* **91**, 4059–4067 (2008).

[4] Usseglio-Viretta, F. L. E. *et al.* Quantitative relationships between pore tortuosity, pore topology, and solid particle morphology using a novel discrete particle size algorithm. *Journal of The Electrochemical Society* **167**, 100513 (2020). URL https://iopscience.iop.org/article/10.1149/1945-7111/ab913bhttps://iopscience.iop.org/article/10.1149/1945-7111/ab913b/meta.

[5] Ahrens, J. P., Geveci, B. & Law, C. C. Paraview: An end-user tool for large-data visualization. In *The Visualization Handbook* (2005). URL https://api.semanticscholar.org/CorpusID:56558637.

[6] Wu, X. & Zhu, Y. Heterogeneous materials: a new class of materials with unprecedented mechanical properties. *Materials Research Letters* **5**, 527–532 (2017).

[7] Lin, G. *et al.* Effect of pore size distribution in the gas diffusion layer adjusted by composite carbon black on fuel cell performance. *International Journal of Energy Research* **45**, 7689–7702 (2021). URL https://onlinelibrary.wiley.com/doi/full/10.1002/er.6350https://onlinelibrary.wiley.com/doi/abs/10.1002/er.6350https://onlinelibrary.wiley.com/doi/10.1002/er.6350.

[8] Cui, C. L., Schweich, D. & Villermaux, J. Influence of pore diameter distribution on the determination of effective diffusivity in porous particles. *Chemical Engineering and Processing* **26**, 121–126 (1989).

[9] Liu, X. *et al.* The influence of pore size distribution on thermal conductivity, permeability, and phase change behavior of hierarchical porous materials. *Science China Technological Sciences* **64**, 2485–2494 (2021). URL http://dx.doi.org/10.1007/s11431-021-1813-0.

[10] Adam, A., Fang, H. & Li, X. Effective thermal conductivity estimation using a convolutional neural network and its application in topology optimization. *Energy and AI* **15**, 100310 (2024). URL http://dx.doi.org/10.1016/j.egyai.2023.100310.

[11] Li, J. X., Rezaee, R., Müller, T. M. & Sarmadivaleh, M. Pore Size Distribution Controls Dynamic Permeability. *Geophysical Research Letters* **48**, e2020GL090558 (2021). URL https://onlinelibrary.wiley.com/doi/full/10.1029/2020GL090558https://onlinelibrary.wiley.com/doi/abs/10.1029/2020GL090558https://agupubs.onlinelibrary.wiley.com/doi/10.1029/2020GL090558.

[12] Tian, S., Ren, W., Li, G., Yang, R. & Wang, T. A Theoretical Analysis of Pore Size Distribution Effects on Shale Apparent Permeability. *Geofluids* (2017). URL https://doi.org/10.1155/2017/7492328.

[13] Kapat, K. *et al.* Influence of Porosity and Pore-Size Distribution in Ti 6 Al 4 V Foam on Physicomechanical Properties, Osteogenesis, and Quantitative Validation of Bone Ingrowth by Micro-Computed Tomography. *ACS Appl. Mater. Interfaces* (2017). URL `www.acsami.org`.

[14] Seibert, P., Raßloff, A., Kalina, K., Ambati, M. & Kästner, M. Microstructure characterization and reconstruction in python: Mcrpy. *Integrating Materials and Manufacturing Innovation* **11**, 450–466 (2022). URL `https://doi.org/10.1007/s40192-022-00273-4`.

[15] Usseglio-Viretta, F. L. *et al.* Matbox: An open-source microstructure analysis toolbox for microstructure generation, segmentation, characterization, visualization, correlation, and meshing. *SoftwareX* **17** (2022).

[16] Stallard, S., Jiang, H., Chen, Y., Bergman, T. L. & Li, X. Exploring the design space of the effective thermal conductivity , permeability , and stiffness of high-porosity foams. *Materials & Design* **231**, 112027 (2023). URL `https://doi.org/10.1016/j.matdes.2023.112027`.

[17] Joyner, L. G., Barrett, E. P. & Skold, R. The determination of pore volume and area distributions in porous substances. ii. comparison between nitrogen isotherm and mercury porosimeter methods. *Journal of the American Chemical Society* **73**, 3155–3158 (1951).

[18] Boerner, T. J., Deems, S., Furlani, T. R., Knuth, S. L. & Towns, J. Access: Advancing innovation: Nsf's advanced cyberinfrastructure coordination ecosystem: Services & support. In *Practice and Experience in Advanced Research Computing*, PEARC '23, 173–176 (ACM, 2023). URL `http://dx.doi.org/10.1145/3569951.3597559`.